$2.00

Reference Card For

MICROSOFT LEVEL III BASIC

MICROSOFT
CONSUMER PRODUCTS
10800 Northeast Eighth, Suite 819
Bellevue, WA 98004

Part No. 10E01
Printed in U.S.A.

SPECIAL CHARACTERS

<break>	Interrupts program execution and returns to BASIC command level
<enter>	Ends every line typed in
←	Erases last character typed
→	Tab (8 columns)
↓	Breaks a logical line into physical lines (linefeed)
Shift@	Toggles output pause switch
Shift←	Erases line being typed
Shift→	Converts display to 32 characters per line
Shift↑	Escape
<clear>	Clears display and converts to 64 characters per line
:	Separates statements typed on the same line
&	Octal constant
&O	Octal constant
&H	Hexadecimal constant

NOTE: Level III statements, commands and functions are printed in red.

VARIABLE TYPE DECLARATION CHARACTERS

			Storage Bytes Used
$	String	(0 to 255 characters)	3 + # of characters
%	Integer	(– 32768 to 32767)	2
!	Single precision	(7.1 digit floating point)	4
#	Double precision	(16.8 digit floating point)	8

SHIFT-KEY ENTRIES

Shift	Sequence	Shift	Sequence
A	AUTO	N	NEXT
B	GET	O	PRINT USING
C	ELSE	P	PUT
D	EDIT <enter>	Q	RETURN
E	EDIT	R	RUN<enter>
F	GOTO	S	SAVE"
G	GOSUB	T	THEN
H	INKEY$	U	TIME$
I	INPUT	V	LOAD"
J	LINE INPUT	W	LEFT$(
K	LINE(	X	MID$(
L	LIST	Y	RIGHT$(
M	LSET	Z	STRING$(

OPERATORS

Symbol	Function
=	Assignment, or equality test
–	Negation or subtraction
+	Addition or string concatenation
*	Multiplication
/	Division (floating point result)
^	Exponentiation

NOT	One's complement (integer)
AND	Bitwise AND (integer)
OR	Bitwise OR (integer)
= <> <= =< >= => <>	Relational test (result is TRUE = −1 or FALSE = 0)

The precedence of operators is:

(1) Expressions in parentheses
(2) Exponentiation (A∧B)
(3) Negation (−X)
(4) ✱, /
(5) +, −
(6) Relational operators (=, <>, <, >, <=, >=)
(7) NOT
(8) AND
(9) OR

ABBREVIATIONS

?	Equivalent to PRINT (Note: L? is not equivalent to LPRINT)
.	Current line for EDIT, NAME, DELETE LIST, LLIST commands
'	Use in place of REM

COMMANDS

Command	Syntax/Function	Example
AUTO	AUTO [first number[,Increment]] *Turn on automatic line numbering*	AUTO 10, 10
CLEAR	CLEAR [n] *Clear program variables. Optional argument sets string space size.*	CLEAR 100
CLOAD	CLOAD "filename" *Load program from cassette tape (Superseded by LOAD in Level III)*	CLOAD "A"
CMD	CMD "T" *Turn off real time clock*	CMD "T"

	CMD "R" *Restart real time clock*	CMD "R"
	CMD "R","mm dd yy hh mm ss" *Set date and time for TIME$*	CMD "R", "02 24 *79 15 04 33*"
CONT	CONT *Continue program execution*	CONT
CSAVE	CSAVE "filename" *Store the resident program on cassette tape* *(Superseded by SAVE in Level III)*	CSAVE "B"
DELETE	DELETE [[start line][-[end line]]] *Delete line(s)*	DELETE 20-50
EDIT	EDIT line number *Edit a program line.* *See EDIT MODE subcommands.*	EDIT 130
LIST	LIST [start line][-[end line]] *List program lines at the terminal*	LIST 100-1000
LLIST	LLIST [start line][-[end line]] *List program lines at the line printer*	LLIST 50-250
LOAD	LOAD "filename" *Load program from cassette tape*	LOAD "B"
NEW	NEW *Delete current program and variables*	NEW
NAME	NAME "new number][,[old number][, increment]] *Renumber program lines*	*NAME 100,,100*
RUN	RUN [line number] *Run a program (from line number)*	RUN RUN 50
SAVE	SAVE "filename" *Store the resident program on cassette tape*	SAVE "X"
SYSTEM	SYSTEM *Enter monitor mode to load machine language file from cassette tape*	SYSTEM
TROFF	TROFF *Turn Trace off*	TROFF
TRON	TRON *Turn Trace on*	TRON

EDIT MODE SUBCOMMANDS (<escape> is shift ↑)

Command	Function
A	Restart EDIT at the start of the line
nCc	Change n character(s)
nD	Delete n character(s) at the current position
E	End editing and save changes but don't type the rest of the line
H string<escape>	Delete the rest of the line and insert string
I string<escape>	Insert string at current position
nKc	Kill all characters up to the nth occurrence of c
L	Print the rest of the line and go to the start of the line
Q	Quit and cancel all changes
nSc	Search for nth occurrence of c
X string<escape>	Go to the end of the line and insert string
n<space>	Move cursor n spaces to the right
n←	Move cursor n spaces to the left
<enter>	End editing and save changes

Note: A string argument can be terminated with <enter>. The effect is equivalent to string<escape> <enter>.

PROGRAM STATEMENTS (except I/O)

Statement	Syntax/Function	Example
DEF	DEF FNx[(argument list)] = expression *Define an arithmetic or string function*	DEF FNA(X, Y) = (X * X * .3 + Y + Y)
	DEF USRn = address *Define the entry address for the nth assembly language subroutine*	DEF USR3 = &2000
DEFDBL DEFINT	DEFtype letter[-letter] *Define double precision,*	DEFDBL A, B, E-G DEFINT I-N

DEFSNG DEFSTR	*integer, single precision, or string default variable names*	DEFSNG X-Z
DIM	DIM variable (size1[,size2...])... *Allocate space for arrays. Arrays may be dimensioned dynamically.*	DIM A(3), B$(10,2,3) DIM Z(2 * I)
END	END *Stop program and return to BASIC command level*	END
ERROR	ERROR *code* *Generate error of code (see table). May call user ON ERROR routine or force BASIC to handle error.*	ERROR 17
FOR	FOR variable=expression TO expression [STEP expression] *Used with NEXT statement to repeat a sequence of program lines. The variable is incremented by the value of STEP.*	FOR I=1 to STEP .5
GOSUB	GOSUB line number *Call a BASIC subroutine by branching to the specified line number. See RETURN.*	GOSUB 210
GOTO	GOTO line number *Branch to specified line number*	GOTO 90
IF	IF expression [THEN] statement [ELSE statement...] *The logical relation in expression is tested. If true, the THEN clause is executed. If false, the ELSE clause is executed.*	IF X<Y THEN Y = X ELSE Y = A
LET	[LET] variable = expression *Assign a value to a variable*	LET X = I + 5 *A* = 44
LSET	LSET RESET *Turn off shift-key entries*	LSET RESET

	LSET SET *Turn shift-key entries back on*	LSET SET
	LSET LIST *List shift-key entries*	LSET LIST
	LSET letter = string expression *Define a new shift-key entry*	LSET M = "MAXIMUM"
NEXT	NEXT variable[,variable...] *Delimits the end of a FOR loop.*	NEXT I
ON ERROR GOTO	ON ERROR GOTO line number *Enables error trap subroutine beginning at specified line. If line number = 0, disables error trapping. If line number = 0 inside error trap routine, forces BASIC to handle error.*	ON ERROR GOTO 1000
ON... GOSUB	ON expression GOSUB line[,line...] *GOSUB to statement specified in list by expression. (If J + 1 = 1, to 20; if J + 1 = 2, to 20; if J + 1 = 3, to 40)*	ON J + 1 GOSUB 20,20,40
ON... GOTO	ON expression GOTO line[,line...] *Branch to statement in list specified by expression. (To 20 if I = 1, to 30 if I = 2)*	ON I GOTO 20,30
OUT	OUT port, byte *Puts byte 16 + I to output port 41.*	OUT 41, 16 + I
POKE	POKE location,byte *Puts byte specified into memory location specified*	POKE 23100,255
RANDOM	RANDOM *Reseeds random number generator*	RANDOM

REM	REM any text *Allows comments in program (not executed). NOTE: ":" does not terminate a REM statement.*	REM TEST X
RESTORE	RESTORE *Resets DATA pointer so DATA statements may be re-read*	RESTORE
RESUME	RESUME or RESUME 0 *Returns from ON...ERROR routine to statement that caused error*	RESUME
	RESUME NEXT *Returns to statement after the one that caused the error*	RESUME NEXT
	RESUME line number *Returns to the specified line*	RESUME 200
RETURN	RETURN *Return from subroutine to statement following last GOSUB performed*	RETURN
STOP	STOP *Stop program execution, print BREAK message, and return to command level*	STOP

INPUT/OUTPUT STATEMENTS

Statement	Syntax/Function	Example
DATA	DATA item[,item...] *Specifies data to be used in a READ statement*	DATA 2.3,"PLUS",4
INPUT	INPUT[#LEN n,m;]["prompt string";]variable [,variable...] *Read data from the terminal. Optional LEN feature branches to line m if no response after n seconds.*	INPUT "VALUES" ;A,B
INPUT#	INPUT#-1,item[,item...] *Input data from cassette*	INPUT#-1,X,Y,B$

LINE INPUT	LINE INPUT[#LEN =m;][“prompt st ng”;]variabl [,variable...] *Assign string variable to line read from the terminal.* *Optional LEN feature branches to line m if no response after n seconds.*	LINE INPUT“TO :N$
PRINT	PRINT *exp*[,*exp*...] *Print data at the terminal*	PRINT A$
	PRINT @ position,exp[,exp...] *Print data at specified position on video display*	PRINT@28, “INDEX”
	PRINT USING “format string”;exp[,exp...] *Print data at the terminal using the format specified. See table for format characters.*	PRINT USING“!”; A$,B$
	LPRINT[USING “format string”;]exp[,exp...] *Print data at the line printer (using the format specified)*	LPRINT A,B
PRINT#	PRINT#-1,item[,item...] *Output data to cassette*	PRINT#-1,J,K,L
	PRINT#-3,item[,item...] *Output data to the RS-232 port*	PRINT#-3,N,T$
READ	READ variable[,variable...] *Read data into specified variables from a DATA statement*	READ I,X,A$

PRINT USING Format Field Specifiers

Numeric

Specifier	Possible Digits	Field Characters	Definition	Example
#	1	1	Numeric field	####
.	0	1	Decimal point	#.#
+	0	1	Print leading or trailing sign. Positive numbers will have “+”, negative numbers will have “–”.	+## ###+

–	0	1	Trailing sign "–" prints ##.##– – if negative, otherwise blank
★★	2	2	Leading asterisk fill ★★###.##
$$	1	2	Floating dollar sign. $$##.## $ is placed in front of the leading digit.
★★$	2	3	Asterisk fill and floating dollar sign. ★★$#.##
,	1	1	Use comma every three digits (left of decimal point only). ##,###.##
^^^^	0	4	Exponential format. Number is aligned so leading digit is non-zero. #.##^^^^

String		
!	Single character	!
/<spaces>/	2 + number of spaces character field	/ /

GRAPHICS STATEMENTS

Statement	Syntax/Function	Example
CLS	CLS *Clear video display*	CLS
GET	GET@ (x1,y1) – (x2,y2)[,G],array name *Save graphics from the video display in an array*	GET@(10,4) – (20,9),A%
LINE	LINE@(x1,y1) – (x2,y2),string expression[,B[F]] *Draw a line, box or filled box using the first character of string expression*	LINE@(10,4) – (20,9)"★",B
	LINE@(x1,y1) – (x2,y2),[RE]SET[,B[F]] *Draw a line, box, or filled box with graphics characters*	LINE@(8,8) – (120,41),SET,B

PUT	PUT@(x1,y1)-(x2,y2)[,action],array name *Put a graphics array on the screen. Action may be SET, RESET, AND, OR, XOR.*	PUT@(1,1)(50,10),SET,A%
RESET	RESET (x,y) *Turns off the graphics block at the given point*	RESET(5,8)
SET	SET(x,y) *Turns on the graphics block at the given point*	SET(63,23)

ARITHMETIC FUNCTIONS

Call	Returns:	Example
ABS(exp)	Absolute value of expression	Y = ABS(A + B)
ATN(exp)	Arctangent of expression (in radians)	PRINT ATN(A)
CDBL(exp)	Double precision representation of expression	A = CDBL(Y)
CINT(exp)	Integer representation of expression	B = CINT(B)
COS(exp)	Cosine of expression (in radians)	A = COS(2.3)
CSNG(exp)	Single precision representation of expression	C = CSNG(X)
EXP(exp)	The constant e to the power of expression	B = EXP(C)
FIX(exp)	Truncated integer of expression	J = FIX(A/B)
INT(exp)	Largest integer representation of expression that is not greater than expression	C = INT(X + 3)
LOG(exp)	Natural logarithm of expression	D = LOG(Y – 2)
RND(exp)	A pseudo-random number. RND(0) returns a single precision value between 0 and 1. RND (integer) returns a pseudo random number between 1 and integer inclusive.	RND(0) RND(100)
SGN(exp)	1 if expression>0 0 if expression = 0 – 1 if expression<0	B = SGN(X + Y)

SIN(exp)	Sine of expression (in radians)	B = SIN(A)
SQR(exp)	Square root of expression	C = SQR(D)
TAN(exp)	Tangent of expression (in radians)	D = TAN(3.14)

STRING FUNCTIONS

Call	Operation	Example
ASC(string)	*Returns the ASCII value of the first character of string*	PRINT ASC(A$)
CHR$(exp)	*Returns a one-character string whose character has the ASCII code of expression*	PRINT CHR$(48)
FRE(string)	*Returns the number of bytes remaining in string space*	PRINT FRE(A$)
INKEY$	*Strobes the keyboard and returns a one-character string corresponding to key pressed during strobe (null string if no key is pressed)*	INKEY$
INSTR([n,]string1,string2)	*Returns the first position of the first occurrence of string2 in string1. Search starts at the nth position.*	INSTR(A$,' ') INSTR(3,X$,Y$)
LEFT$(string,length)	*Returns leftmost length characters of the string expression*	B$ = LEFT$ (X$,8)
LEN(string)	*Returns the length of string (0 for null string)*	LEN(A$ + B$)
MID$(string,start[,length])	*Returns characters from the middle of the string starting at the position specified to the end of the string of for length characters*	A$ = MID$(X$,5,10)

MID$(string1,n,m) = string2
On the left side of an equation, replaces a portion of string1 with string2, starting at position n, for m characters — MID$(K$,1,8) = L$

RIGHT$(string,length)
Returns rightmost length characters of the string expression — C$ = RIGHT$(X$,8)

STR$(exp) — *Returns a string representation of the numeric expression* — PRINT STR$(35)

STRING$(count,string)
Returns a string count long containing first character of string — X$ = STRING$(10,"A")

STRING$(count,exp)
Returns a string count long containing characters with ASCII code of numeric expression — Y$ = STRING$(100,42)

TIME$ — *Returns a string of form mm/dd/yy hh:mm:ss that represents the elapsed time since Level III BASIC was brought up* — PRINT TIME$

VAL(string) — *Returns the numeric value of the string representation of a number* — PRINT VAL("3.1")

SPECIAL FUNCTIONS

Function	Operation	Example
ERL	Returns line number of current error	PRINT ERL
ERR/2 + 1	Returns code number of current error	PRINT ERR/2 + 1
FRE(number)	Returns number of unused bytes in memory	FRE(0)
INP(port)	Returns the byte read from the specified input port	PRINT INP(21)

MEM	Returns number of unused bytes in memory	MEM
PEEK(add)	Returns the byte read from the specified address	PRINT PEEK (&2000)
POINT(x,y)	Returns −1 if specified graphics block is "on." Returns 0 if specified graphics block is "off." 0<=x<128 and 0<=y<48	POINT(20,10)
POS(dummy)	Returns current position of cursor	POS(0)
TAB(exp)	Tabs cursor to specified position. Used only in PRINT statements.	PRINT TAB(20);X$
USRn(arg)	Branches to machine language subroutine	USR2(Y)
VARPTR(var)	Returns address of variable in memory	I = VARPTR(X)

ERROR MESSAGES

Code	Abbreviation	Error Message
1	NF	NEXT without FOR
2	SN	Syntax error
3	RG	Return without GOSUB
4	OD	Out of data
5	FC	Illegal function call
6	OV	Overflow
7	OM	Out of memory
8	UL	Undefined line
9	BS	Subscript out of range
10	DD	Redimensioned array
11	/0	Division by zero
12	ID	Illegal direct
13	TM	Type mismatch
14	OS	Out of string space
15	LS	String too long
16	ST	String formula too complex
17	CN	Can't continue
18	NR	No RESUME
19	RW	RESUME without error
20	UE	Unprintable error
21	MO	Missing operand
22	FD	Bad file data
23	L3	Disk BASIC only